Варіант 7

Умова:

Розв'язати крайову задачу

$$Ay = f; \quad x_1 < x < x_2;$$
$$a_1 y = f_1; \quad x = x_1;$$
$$a_2 y = f_2; \quad x = x_2;$$

де $Ay = y'' + p(x)\,y' + q(x)\,y = f(x);$ $p(x), q(x)$ - задані функції; a_1, a_2 - задані оператори; x_1, x_2 - задані числа.

Для дослідження точності методів потрібні задачі, що мають аналітичний розв'язок. Тому розв'язком Вашої задачі є функція

$$y(x) = ax^2 + bx + c + (dx+e)^{-1}.$$

Допрограмовий етап:

$$y = 0.267\,x^2 + 0.871\,x - 0.194 + (-1.093\,x - 1.572)^{-1}$$

$$Ay = y'' - 3\,y' + \frac{y}{x}$$

$$y' = 0.534\,x - 0.871 - 1.093 * (-1.093\,x - 1.572)^{-2}$$

$$y'' = (y')' = 0.534 + 2.389298 * (-1.093\,x - 1.572)^{-3}$$

$$\frac{y}{x} = 0.267\,x + 0.871 - \frac{0.194}{x} - \frac{1}{x} * (-1.093\,x - 1.572)^{-1}$$

$a_1\,y = y(0.4) = -0.300590531555$ крайова умова типу Діріхле

$a_2\,y = y(0.7) - 2\,y'(0.7) = 3.000846632502$ змішана крайова умова

Текст програми:

```
import numpy as np
import matplotlib.pyplot as plt

A = 0.267; B = 0.871; C = -0.194; D = -1.093; E = -1.572
P = -3.; Q = lambda x: 1./x

def y(x):
    global A; global B; global C; global D; global E
    return A*pow(x, 2) + B*x + C + 1./(D*x + E)

def dy(x):
    global A; global B; global D; global E
    return 2.*A*x + B - D/pow((D*x + E), 2)

def d2y(x):
    global A; global D; global E
    return 2.*A + 2.*pow(D, 2)/pow((D*x + E), 3)

def f(x):
    global P
    return d2y(x) + P*dy(x) + Q(x)*y(x)

def fdMethod(n=30, x0=0.4, xn=0.7):
    h = float((xn - x0)/n)
```

```
    x = np.zeros(n + 1); x[0] = x0
    for i in range(1, n + 1):
        x[i] = x[i - 1] + h

    y_ = np.zeros(n + 1)
    for i in range(n + 1):
        y_[i] = y(x[i])

    matr = np.array([[0.]*(n + 1)]*(n + 1))
    matr[0, 0] = -2./pow(h, 2) + Q(x[0]) - 3./h + 1.5*P
    matr[0, 1] = 2./pow(h, 2)
    matr[n, n] = 2.
    for i in range(1, n):
        matr[i, i - 1] = 1./pow(h, 2) - P/(2*h)
        matr[i, i] = -2./pow(h, 2) + Q(x[i])
        matr[i, i + 1] = 1./pow(h, 2) + P/(2*h)

    b = np.array([0.]*(n + 1))
    b[0] = f(x[0]) - 2.*(1.5*y(x0) - dy(x0))/h + P*(1.5*y(x0) - dy(x0))
    b[n] = 2.*y(xn)
    for i in range(1, n):
        b[i] = f(x[i])

    return x, np.linalg.solve(matr, b), y_, h

N = 10
BEST = 4
n = (30, 40, 60, 80, 100, 150, 300, 400, 500, 3000)

x_ = [[0.]]*N
y_fd = [[0.]]*N
y_ = [[0.]]*N
h = np.zeros(N)
for i in range(N):
    x_[i], y_fd[i], y_[i], h[i] = fdMethod(n=n[i])

fdAcc = [[0]]*N
for i in range(N):
    fdAcc[i] = abs(y_[i] - y_fd[i])

hfdAcc = np.zeros(N)
for i in range(N):
    hfdAcc[i] = np.amax(fdAcc[i])


print "x           y               y_fd              Acc"
for i in range(n[BEST] + 1):
    print x_[BEST][i], "   ", y_[BEST][i], "  ", y_fd[BEST][i], "    ", fdAcc[BEST][i]

print "n       h                   e(h)"
for i in range(N):
    print n[i], "    ", h[i], "     ", hfdAcc[i]


plt.plot(x_[BEST], y_[BEST], x_[BEST], y_fd[BEST])
plt.title("y and y_fd for n = %i"%n[BEST])
plt.grid(True)
plt.xlabel('x')
plt.ylabel('y')
```

```
plt.show()
plt.plot(x_[BEST], fdAcc[BEST])
plt.title("Residual for n = %i"%n[BEST])
plt.grid(True)
plt.xlabel('x')
plt.ylabel('fdAcc')
plt.show()

plt.plot(h, hfdAcc)
plt.title("Residual max for all n")
plt.grid(True)
plt.xlabel('h')
plt.ylabel('hfdAcc')
plt.show()
```

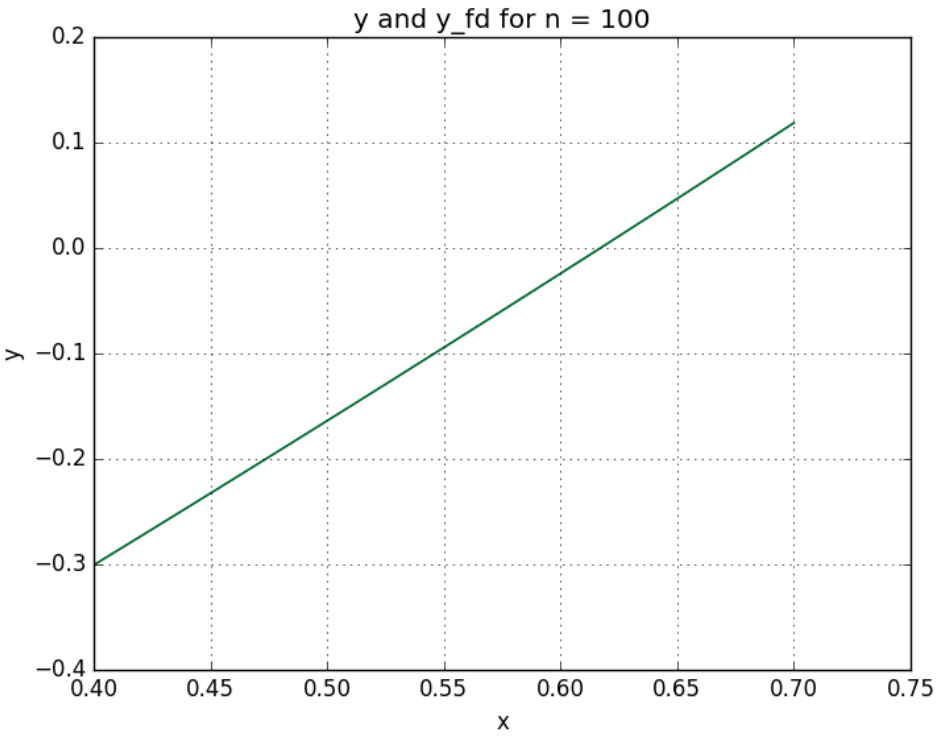
Результати роботи програми:

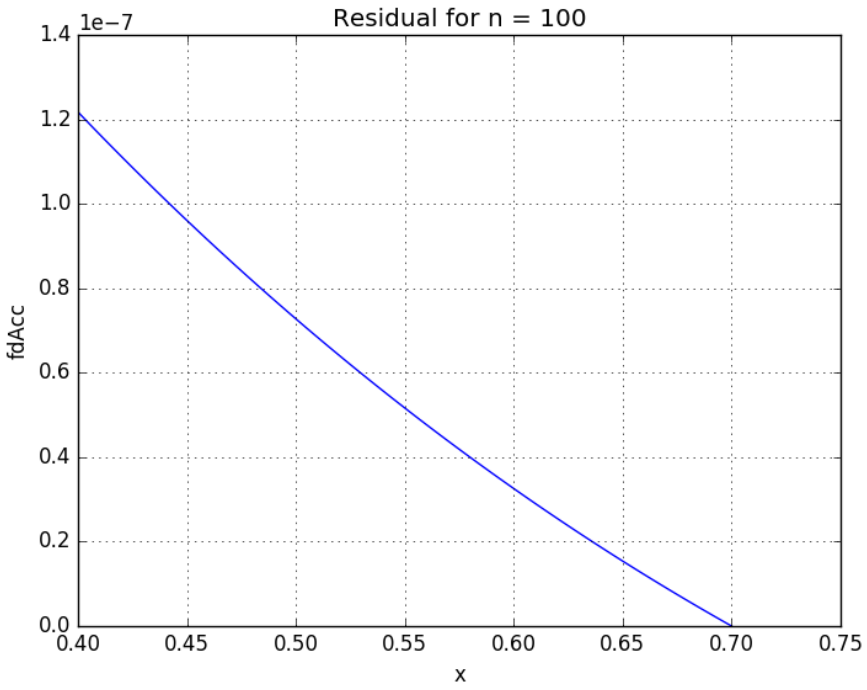
x	y	y_fd	Acc
0.4	-0.300590531555	-0.300590409821	1.21733362279e-07
0.403	-0.296523391977	-0.296523271855	1.20122409075e-07
0.406	-0.292454084674	-0.292453966153	1.18520664172e-07
0.409	-0.28838259679	-0.288382479862	1.16928082827e-07
0.412	-0.284308915556	-0.284308800211	1.1534462091e-07
0.415	-0.280233028283	-0.280232914513	1.13770234289e-07
0.418	-0.276154922366	-0.276154810161	1.12204879388e-07
0.421	-0.27207458528	-0.272074474632	1.10648512353e-07
0.424	-0.267992004581	-0.26799189548	1.09101089885e-07
0.427	-0.263907167906	-0.263907060343	1.07562568463e-07
0.43	-0.259820062969	-0.259819956937	1.06032905123e-07
0.433	-0.255730677566	-0.255730573054	1.04512057009e-07
0.436	-0.251638999569	-0.251638896569	1.02999981599e-07
0.439	-0.247545016928	-0.247544915432	1.01496636207e-07
0.442	-0.24344871767	-0.243448617668	1.00001978892e-07
0.445	-0.239350089898	-0.239349991382	9.85159673561e-08
0.448	-0.235249121792	-0.235249024753	9.7038560215e-08
0.451	-0.231145801604	-0.231145706034	9.55697156413e-08
0.454	-0.227040117664	-0.227040023555	9.41093924456e-08
0.457	-0.222932058375	-0.222931965717	9.26575496607e-08
0.46	-0.218821612211	-0.218821520997	9.12141465415e-08
0.463	-0.214708767722	-0.214708677943	8.97791425369e-08
0.466	-0.210593513528	-0.210593425176	8.83524974848e-08
0.469	-0.206475838321	-0.206475751387	8.69341713339e-08
0.472	-0.202355730865	-0.202355645341	8.55241243103e-08
0.475	-0.198233179993	-0.198233095871	8.41223171399e-08
0.478	-0.194108174609	-0.19410809188	8.27287104377e-08
0.481	-0.189980703686	-0.189980622343	8.13432654845e-08
0.484	-0.185850756266	-0.1858506763	7.99659436168e-08
0.487	-0.181718321459	-0.181718242862	7.85967063655e-08
0.49	-0.177583388443	-0.177583311208	7.72355157608e-08
0.493	-0.173445946464	-0.173445870581	7.58823340274e-08
0.496	-0.169305984832	-0.169305910295	7.45371236677e-08
0.499	-0.165163492927	-0.165163419727	7.3199847378e-08
0.502	-0.161018460192	-0.161018388322	7.18704682712e-08
0.505	-0.156870876137	-0.156870805588	7.05489498487e-08

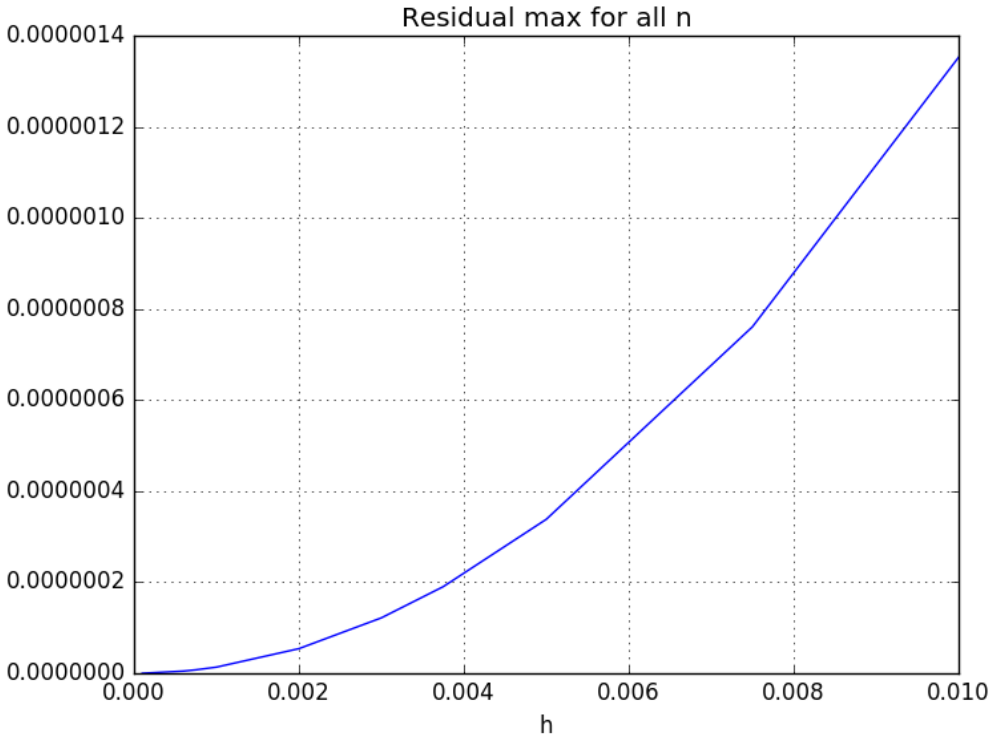
0.508	-0.152720730334	-0.152720661099	6.9235255723e-08
0.511	-0.148568012422	-0.148567944493	6.79293497285e-08
0.514	-0.144412712102	-0.144412645471	6.66311961439e-08
0.517	-0.14025481914	-0.140254753799	6.53407594697e-08
0.52	-0.136094323362	-0.136094259304	6.40580047617e-08
0.523	-0.131931214659	-0.131931151876	6.27828969368e-08
0.526	-0.127765482982	-0.127765421466	6.15154014394e-08
0.529	-0.123597118343	-0.123597058088	6.02554840745e-08
0.532	-0.119426110817	-0.119426051814	5.90031107445e-08
0.535	-0.115252450537	-0.115252392779	5.77582478373e-08
0.538	-0.111076127698	-0.111076071177	5.65208620878e-08
0.541	-0.106897132552	-0.106897077261	5.52909203977e-08
0.544	-0.102715455412	-0.102715401344	5.40683899181e-08
0.547	-0.0985310866492	-0.098531033796	5.28532381194e-08
0.55	-0.0943440166924	-0.0943439650469	5.16454328886e-08
0.553	-0.0901542360282	-0.0901541855833	5.04449425148e-08
0.556	-0.0859617352008	-0.0859616859491	4.92517351486e-08
0.559	-0.0817665048109	-0.0817664567451	4.80657798563e-08
0.562	-0.0775685355155	-0.0775684886285	4.68870454684e-08
0.565	-0.073367818028	-0.0733677723125	4.57155016065e-08
0.568	-0.0691643431169	-0.0691642985658	4.455111767e-08
0.571	-0.0649581016061	-0.0649580582122	4.33938636413e-08
0.574	-0.0607490843741	-0.0607490421304	4.22437099329e-08
0.577	-0.0565372823537	-0.0565372412531	4.11006269505e-08
0.58	-0.0523226865318	-0.0523226465672	3.99645857241e-08
0.583	-0.0481052879486	-0.048105249113	3.88355573464e-08
0.586	-0.0438850776975	-0.0438850399839	3.77135132637e-08
0.589	-0.0396620469245	-0.0396620103261	3.65984253667e-08
0.592	-0.0354361868282	-0.0354361513379	3.54902654487e-08
0.595	-0.0312074886589	-0.0312074542699	3.43890063752e-08
0.598	-0.0269759437187	-0.026975910424	3.32946204565e-08
0.601	-0.0227415433605	-0.0227415111534	3.22070808981e-08
0.604	-0.0185042789884	-0.018504247862	3.11263608672e-08
0.607	-0.0142641420567	-0.0142641120043	3.00524340257e-08
0.61	-0.0100211240699	-0.0100210950846	2.89852744619e-08
0.613	-0.00577521658213	-0.00577518865728	2.79248559849e-08
0.616	-0.00152641119688	-0.00152638432573	2.6871153384e-08
0.619	0.0027253004334	0.00272532625754	2.58241413869e-08
0.622	0.00697992660763	0.00697995139143	2.47837951464e-08
0.625	0.0112374755765	0.0112374993266	2.37500901099e-08
0.628	0.0154979555426	0.0154979782656	2.27230018083e-08
0.631	0.0197613746613	0.0197613963638	2.17025064274e-08
0.634	0.0240277410404	0.024027761729	2.06885801442e-08
0.637	0.0282970627411	0.0282970824223	1.96811997463e-08
0.64	0.0325693477777	0.032569366458	1.86803418845e-08
0.643	0.0368446041187	0.0368446218046	1.76859838302e-08
0.646	0.0411228396864	0.0411228563845	1.66981032959e-08
0.649	0.0454040623577	0.0454040780744	1.57166778411e-08
0.652	0.0496882799643	0.049688294706	1.47416856638e-08
0.655	0.0539755002929	0.053975514066	1.37731050107e-08

0.664	0.0668552548119	0.0668552657175	1.09056217529e-08
0.667	0.0711545630095	0.071154572972	9.96247738472e-09
0.67	0.0754569122006	0.0754569212263	9.02564137317e-09
0.673	0.0797623099093	0.0797623180044	8.09509329014e-09
0.676	0.0840707636169	0.0840707707877	7.17081405366e-09
0.679	0.0883822807623	0.0883822870151	6.25278402666e-09
0.682	0.0926968687423	0.0926968740833	5.34098441862e-09
0.685	0.097014534912	0.0970145393474	4.43539653616e-09
0.688	0.101335286585	0.101335290121	3.53600201897e-09
0.691	0.105659131033	0.105659133676	2.64278254836e-09
0.694	0.109986075488	0.109986077244	1.75572055505e-09
0.697	0.114316127141	0.114316128015	8.74798775086e-10
0.7	0.118649293141	0.118649293141	5.55111512313e-16

n	h	e(h)
30	0.01	1.35258589778e-06
40	0.0075	7.60831526003e-07
60	0.005	3.38147957024e-07
80	0.00375	1.902083821e-07
100	0.003	1.21733362279e-07
150	0.002	5.41035795609e-08
300	0.001	1.35255708855e-08
400	0.00075	7.60714402581e-09
500	0.0006	4.86952173029e-09
3000	0.0001	8.36148927874e-11







Висновки:

Цей метод дає гарні результати (похибка < 1E-5) на малих проміжках та при не надто малій кількості ітерацій. Під час виконання роботи були виявлено, що похибка отриманих результатів параболічно залежить від величини ітераційного інтервалу з коефіцієнтом a~0.0130092745497 (e(h)~a*h*h) .